Briana Long
8/17/2025
Module 7

Slide Show transcript

Slide 2
Hello! My name is Briana Long. I am a senior Computer Science major at SNHU.
This presentation is to highlight migrating a full-stack application to Amazon Web Services. This is a serverless, cloud-based environment. This pivot allows for cost-effective scalability. While also integrating security and automation.

Slide 3
Monolithic architecture is a software design where all components of an application are integrated and deployed as a single unit.
Compared to using Docker, which is a microservice that breaks down components into smaller services.
It is used as a container for consistent environments across machines.

Slide 4
Orchestration is the automation and management of complex tasks and workflows across systems and services.
It creates a central point of control to streamline the process and reduce manual intervention.
Docker Compose runs multiple containers in sync. This includes frontend, backend, and a database. This reduces setup errors that can occur.

Slide 5
Rather than no servers, AWS manages the servers.
S3 storage is a cloud-based storage service provided by Amazon Web Services.
Data, known as objects, is stored in buckets rather than locally. This makes the data more accessible compared to local storage on one machine.
The key features of using a serverless cloud are scalability, availability, security, and cost effectiveness.
This makes it the perfect solution for startups that want to scale up.

Slide 6
The serverless API is used to route requests from the frontend to the Lambda functions.
The code runs only when there are requests, allowing for a cost-effective server.
The front end is exposed through REST endpoints that use the API Gateway.

The front end makes HTTP requests while Lambda executes backend logic and response.
The Lambda function contains the backend logic. A script is made and run in Node.js.
It connects to the DynamoDB and returns a JSON response.

Slide 7
MongoDB uses flexible JSON documents, queries, and secondary indexes
DynamoDB uses key-value and documents, PK/SK schema, access pattern-driven, and GSIs
Queries used were GET, OPTIONS, POST, PUT, and DELETE.
Scripts were produced and used in queries that produced a JSON item.
Using DynamoDB is a serverless, flexible access patterned model compared to a documented one.
Every item has a partition key and a sort key in tables.

Slide 8
Elasticity of services that scale with traffic and prevent idle servers.
This is important when using a serverless design that scales automatically.
Pay-for-use model is billed per request, storage, or reads. This allows for accurate costs that align with usage.
Equals faster features, lower overhead costs, and predictable costs, and scale and user increases occur.
The graph illustrates that traditional setups waste money and hurt customers compared to a cheaper serverless setup.

Slide 9
For access control, we authenticate users with Cognito, which issues JWTs the API validates either natively or through a Lambda Authorizer.
All traffic uses HTTPS with an ACM certificate and a custom domain.
We put AWS WAF in front of API Gateway for IP allow/deny and common attack protection, and we enabled throttling and usage plans to limit abuse.
API Gateway also validates request bodies against JSON Schema before requests ever reach Lambda.
Data at rest is encrypted with KMS, and application secrets live in Secrets Manager/Parameter Store.

Slide 10
For policies, we followed least-privilege.
A Lambda execution role gets only the DynamoDB actions it needs on our table and GSIs, scoped to ARNs.
Another policy limits S3 access to a specific bucket prefix, and we grant the minimal KMS

permissions required.
API Gateway has permission to write logs.

Slide 11
Gateway/Lambda path verifies JWTs and enforces CORS.
The Lambda-to-DynamoDB path uses IAM—no embedded credentials.
For S3, we block public access and use pre-signed URLs so clients never get blanket bucket permissions.
This combination prevents unauthorized access

Slide 12
Serverless is the best choice for startups. It is a cost effective solution compared to containers.
Data and integration allows LAMBDA to answer core queries. An access pattern is used to increase infrastructure.
Security is a feature by default. This prevents unauthorized users, least privilege policies in place along side encryption and end to end HTTPS.